1 Aim

This KISS device (keep it simple, stupid) generates the pulses for motor ignition tester in a particularly nasty environment. The output frequency covers the range 5 to 200Hz, this means :

- 150 to 6000 rpm for a 4cyl / 4 stroke engine
- 300 to 12000 rpm for a 2cyl / 4 stroke, or for a 1 cyl / 2 stroke

It is made using a small Digispark module Tiny85.
The power supply comes from the very same 12V battery used for ignition.

An analog display of the frequency is made by a needle galvanometer.
The « **dwell** » (historic mechanical way of insuring the magnetization time of the coil, the real need) is permanently **set to 10ms**, it automatically restricts itself to **4ms** @ 200Hz, and guarantees a **spark time of 1ms** minimum.
A new version, based on an arduino nano USB board, gives more accurate results, if really needed.

2 Environment

The lab environment is even worse than the engine compartment

- the generator is located less than 30cm from the spark
- the generator box can work with opened box
- the spark wire is not resistive nor shielded
- the spark plug has no resistance (according to today's standard, the internal resistor is around 5kΩ)
- the spark plug is a pair of electrodes in air (using Paschen's law, 8mm air gap represents 0,8mm in typical 4 stroke engine, or 1,25mm in 2 stroke engine)
- the spark plug has no shielding at all, in a real engine, there is an absolute minimum of shielding (cylinder + cylinder head)

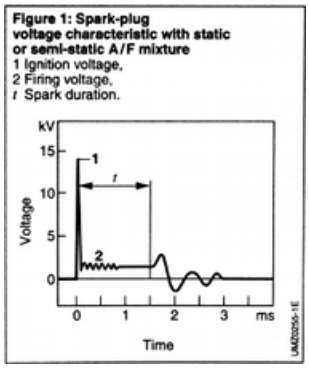
Because of that environment, I blew up 4 microcontroller boards during development ! that's why I became paranoid in protecting the generator.
It could now be considered as bulletproof, as well as the ignition module, used since years for various developments. The ignition module is described on the french part of my site (http://www.hackerschicken.eu/www/electric/commande_allumage.pdf) It uses bipolar transistors only, more reliable as the « specifically developped » MOS or IGBT, and it seems the car manufacturers use now a current controlled coils scheme, and the MOS are particularly unable to work in linear (and harsh) world. Stopping the real development and maturing of those « specifically developped » MOS or IGBT. The ignition modules restrict the coil current to 3,9A and can control almost any coil, from 0,4Ω to 5Ω.
The destructions came obviously through the conducted perturbation path, more than through the radiated perturbation path, as it is possible to run it reliably with box opened (not advised, though).

3 The spark

The aim of this device is to generate a spark by the mean of a driver, which manages the current and voltage of the coil, while the dwell time and the spark duration are managed by this generator. The spark, once ignited, has a minimum duration, around 1,5ms for a car engine, sometimes shorter for a small 2-stroke engine.
The spark duration is fixed at 1ms in the tiny85 version. The arduino nano version has adjustable duration and dwell time.

typical spark

source : Bosch Automotive Handbook



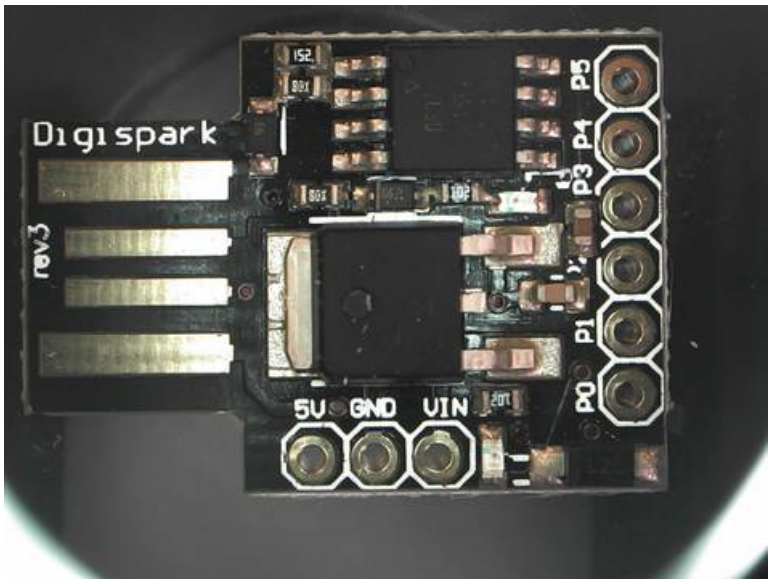
4 The microcontroller board (µC)

It is an « arduino » board. I just use it as a cheap atmel microcontroller ($1.5), no use of arduino environment, nor of the USB connection. Program written in plain vanilla C (avr-gcc in Linux environment) loaded through ICSP port.
The processor used here is an atmel Tiny85 in 8-pin package, there are 5 pins left to interface to the world. The PB5 pin pin is reserved for reset (if you absolutely need it, you can no longer use the standard programming scheme.Not a big choice is possible for distributing the pins because all the possible functions are preset to certain pins only. You better read the datasheet (RTFM) to use that chip !

The digi**spark** board

well-named for this use !

27 x 19 mm

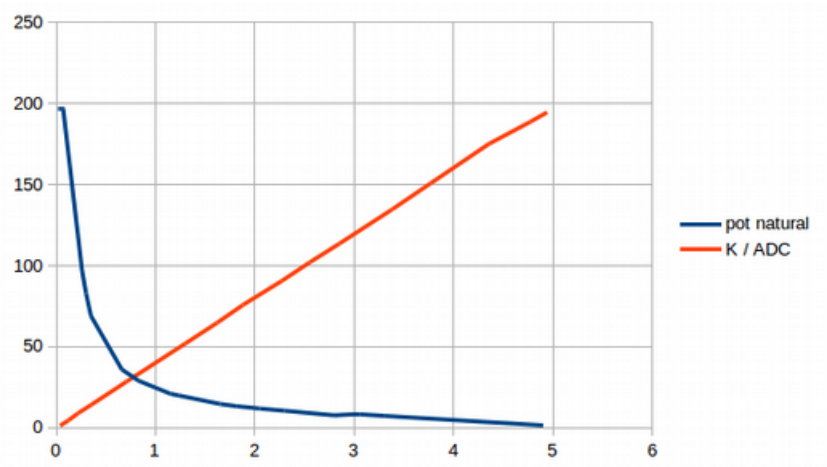


The timer1 (8-bit, on PB0 pin) is used for a PWM 0 to 5V for the analog display. The frequency is not very important, as it will be integrated by the moving coil and the needle of the galvanometer (actually around 250Hz). The ratio representing the rpm, is generated from the corresponding internal variable.
The timer2 (8-bit) is used as the time base, by generating the ticks, adjusted at 10kHz for en easier development. The processor uses its internal RC oscillator, with his poor accuracy, barely enough here. The internal system clock is set to 8MHz by fuse setting (lfuse = 0xFF). We can now use the ticks as a x-bit timer for the rpm creation (actually around 14-bit).
The « waveform » function of the PWM is created for the « dwell » (remember, it's only a time, for energizing the coil inductance) and available on PB1 pin
This waveform can also provide a spark time of 1ms minimum, giving enough time to inflamate the gas.

An analog input to the internal A/D converter reads the potentiometer value for the frequency control. The potentiometer is in ratiometric mode : voltage divider of the Vcc voltage, it is thus not affected by the potentiometer tolerance, nor by Vcc variation, it just needs to be a linear pot.

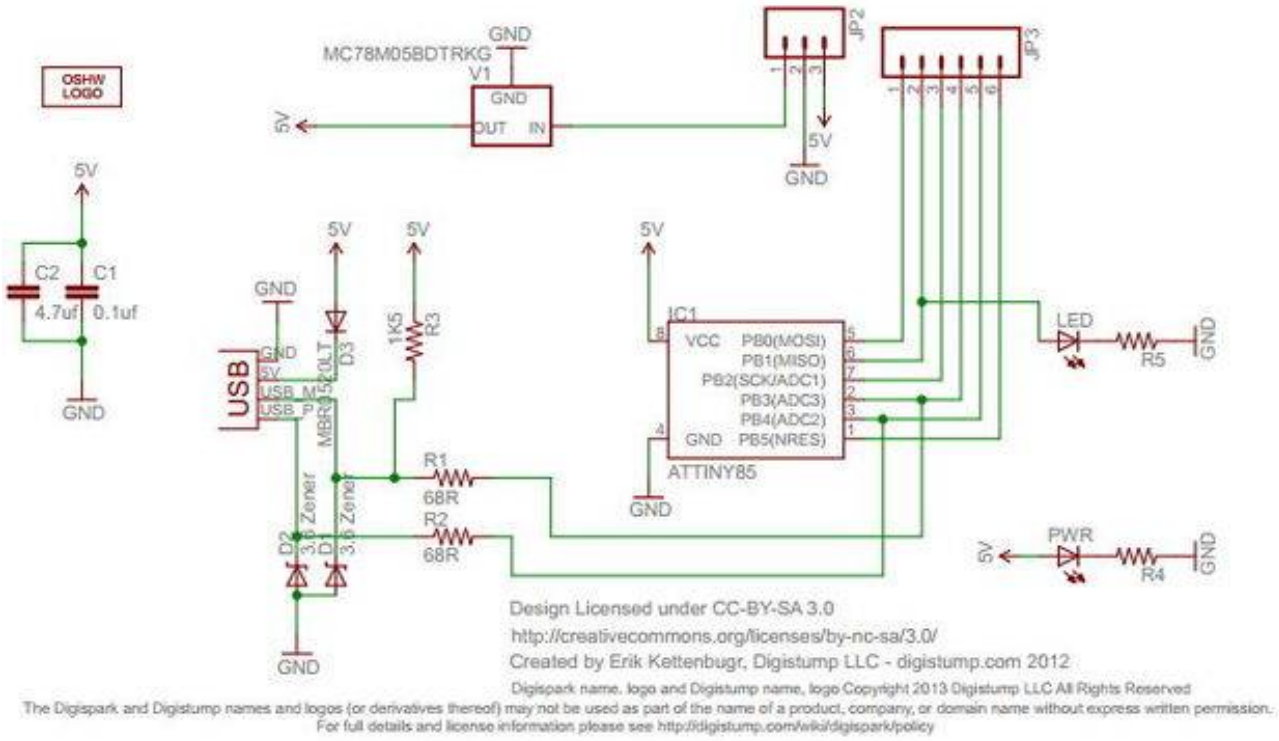
The old fashion of frequency generators used a monostable multivibrator (555 or HC123). They are not really linear in nature (presence of a constant ON time, in addition to the cyclic ratio), but the frequency follows more ore less a function of the resistance giving an apparent linearity in behaviour freq = f(rotation angle)
The µC fashion controls the **period** of the frequency. The feeling is not nice : the major change in frequency is located in a few angle of rotation of the knob ! (blue trace)

A small change in software makes the behaviour becoming now : freq = k * 1/period. No need for a LUT (Look-Up Table). You only have to invert the GND and Vcc pins on the pot to have min to the left and max to the right.



The internal re-inversion of the ADC reading regenerates a linear behaviour (red trace) nice feeling now. It acts as a linear VCO (voltage controlled oscillator).

Schematics of the Digispark



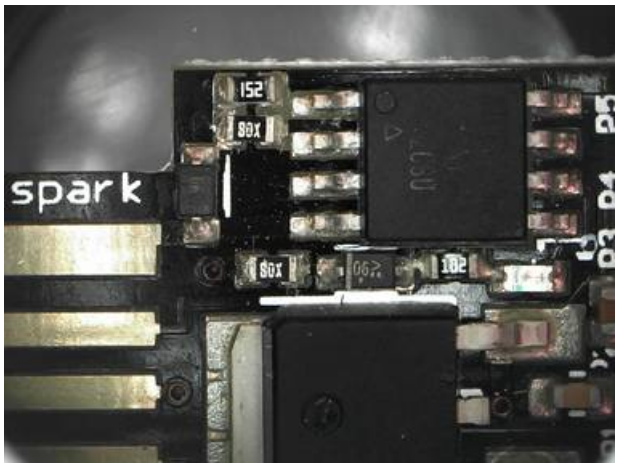
Another feature of the ADC use : I wanted first to stay compatible with the pinout of a previous version of the generator made for a friend (two pots, one for rpm, the other for dwell, now unuseful). I used the ADC2 & ADC3 inputs. But they are already used in the Digispark for the USB link.
Those ADC used for USB in the Digispark have a very specific connection : series resistor of 68Ω connected to a 3,6V Zener diode. Even with a linear pot, the response is completely screwed !

The ADC conversion speed is a SAR A/D converter (Successive Approximation Register). It begins to evaluate the MSB Most Significant Bit). The max allowed clock speed is 200kHz, if

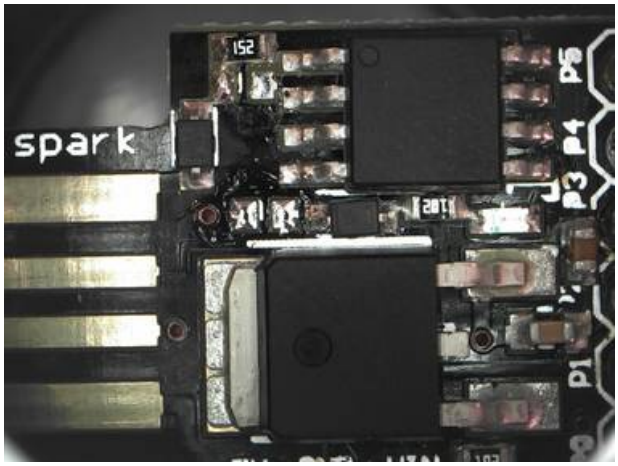
you go over this limit, you begin to lose the LSB (Less Significant Bit), as the LSB correct reading is not mandatory. Clock of ADC is 62kHz.

I then modified the analog input (I only need one now, ADC1, as the dwell is computed by the software). **I use now ADC1 / PB2 for the pot**.

Board modification, just in case you need a free use of PB3 - PB4



With 68Ω resistors



Without 68Ω resistors

With a standard soldering iron, flat round end 3mm, I got rid of the 68Ω resistors (stamped 50X ! and measuring 68Ω) This eliminates the USB connection, programmation is still possible through the ICSP 6-pin port. The USB track could even be dremeled out ! (not yet tested)
This modification is only needed if you intend to use one or both of the ADC2 – ADC3 port. The actual tester uses only one potentiometer, so it has to be ADC1 (PB2), and no modification on USB ports.

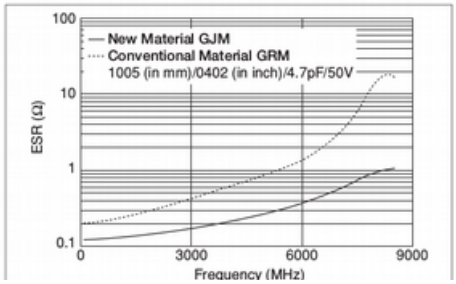
5 the board protection

The major perturbations come by conducting path, so I protected the possible parasitic inputs to the board : power supply, and output to ignition module. The pot, as powered by a filtered Vcc, and internal to the case, needs nothing specific. The voltmeter display is energized by the filtered board, but, as they are wires traveling outside, I prefer to add a mild protection with 2 Schottky diodes to GND and Vcc.

The board is powered by the 12V battery. The protection uses a 10nF bypass capacitor TUSONIX 4400-035 C FILTER (also called bypass, or feedthrough filter, or EMI filter) at the passing of the wall.
Then comes a ferrite bead (maybe superfluous, but I was paranoid after destruction of boards)
Followed by a ferrite inductance and capacitors to the ground (470µF/25V and ceramic 0,1µF)
The internal 78M05 gives the internal Vcc of 5V, and withstands 35V on input.

The feedthrough filter is very efficient at high frequencies specifically with a low impedance power supply like a lead-acid battery, compared to a decoupling capacitor, which has only a low ESR to oppose, in a voltage divider, to that source.

ESR of a ceramic capacitor like Murata GRM55DR72J224KW01 :

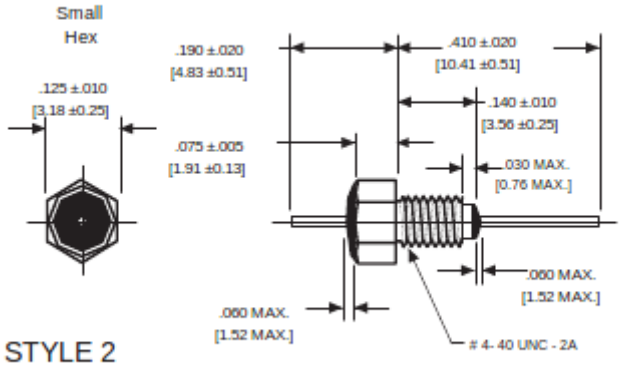


at 1GHz the ESR of the Murata is about 0,1Ω
when placed at the entrance of the box, and powered by a lead-acid battery, will divide the HF by only 2 (**6dB**)

Tusonix filters :

TUSONIX Part Number	Style	Lead Dia.	Circuit	Working Voltage (dc) 85°C	Working Voltage (dc) 125°C	Capacitance (pF)	Capacitance Tolerance	Current (A)	Minimum No-Load Insertion Loss (dB) at 25°C per MIL-Std-220 1MHz	10MHz	100MHz	1GHz	10GHz
Pi Configuration													
4261-001	1	.018 [.46]	Pi	--	50	5500	GMV	3	--	14	55	70	--
4200-012	1	.018 [.46]	Pi	--	200	1500	-0,+100%	3	--	5	42	70	--
C Configuration													
4403-035	2	.030 [.76]	C	70	50	10,000	GMV	10	4	21	35	50	60

The bypass EMI filter divides the HF 1GHz energy by 316 (**50 dB**), regardless of the source impedance ! Under around 1MHz, the ESR of the Murata protects better, while over that, the bypass is much better.



The Tusonix bypass used

The **interface transistor** is mounted outside of the metal box and because it is a BJT, has an important parasitic capacitance (Miller, or collector-base cap, few manufacturers confess it in the datasheets). The signal penetrates the wall trough the same bypass as for power supply.
Then comes a ferrite bead
Followed by a ferrite inductance and a 0,1µF to ground
There is no 78M05 for filtering the ignition pulse, so I clamped the line to GND and Vcc with Schottky diodes, and a Zener 5,1V to ground.

Paranoid, I said, but after proven destructions !
Remember, the spark speed is in the range of 100 000 km/s (needs 1 millionth of a millionth of a second to travel the spark gap, with corresponding perturbation spectrum)

it is now bulletproof.

Internal protection of the µC (from datasheet)

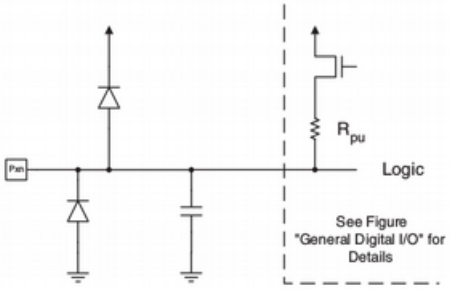
the µC has protection to all I/O pins, including ADCs, by clamping diodes

10.1 Introduction

21.1 Absolute Maximum Ratings*

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 10-1. Refer to "Electrical Characteristics" on page 166 for a complete list of parameters.

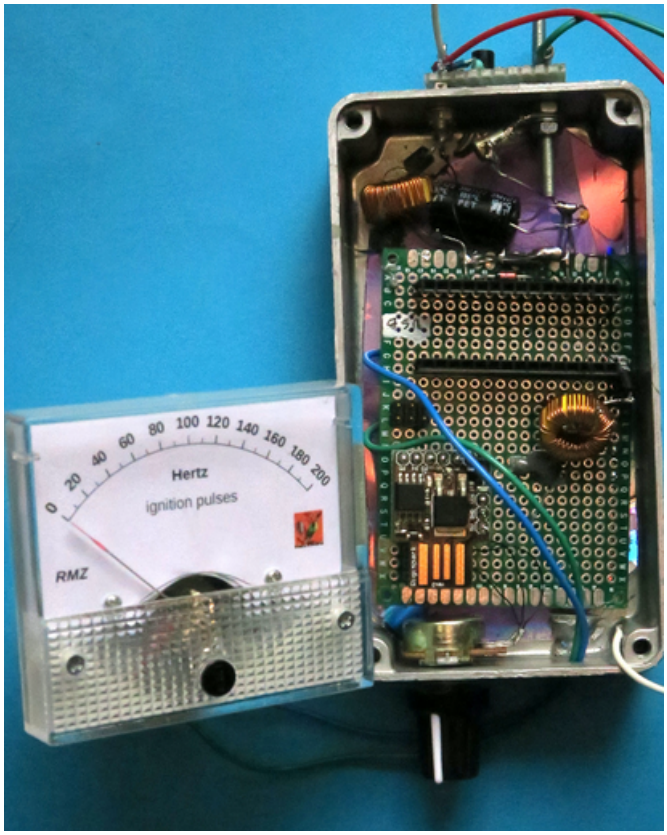
Figure 10-1. I/O Pin Equivalent Schematic



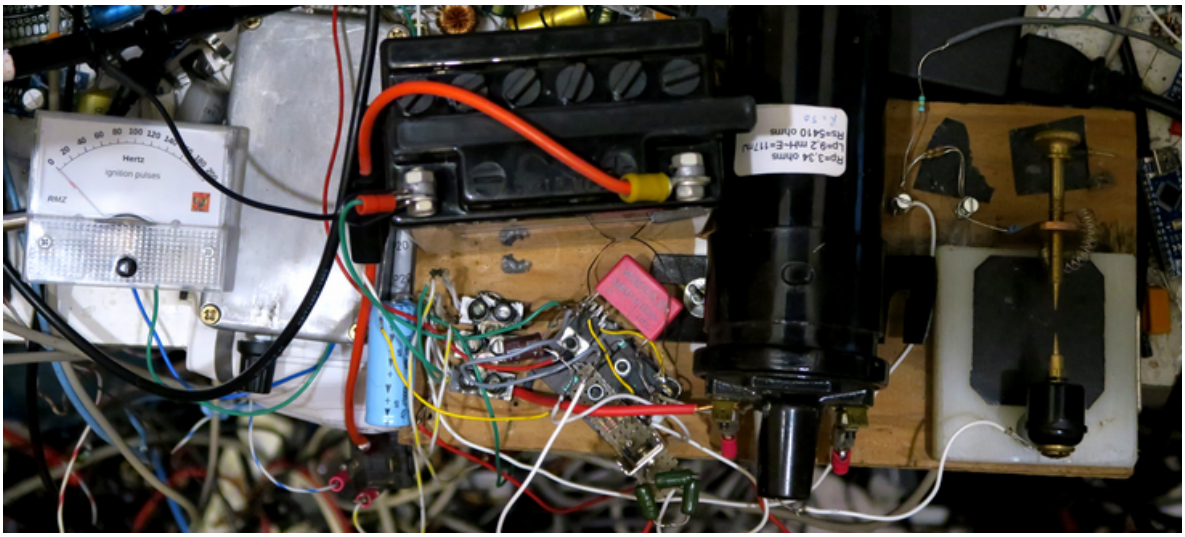
the absolute maximum of 0,5V indicates that the internal clamping diodes ARE NOT of Schottky type because at 0,5V, the current of a Schottky would be in the range of 10A ! See http://www.hackerschicken.eu/www/electric/diode_moto.pdf

An external diode **should be of Schottky type**, and conducts at lower voltage than the internal clamping. This gives a good protection to the µC.

6 Final hardware

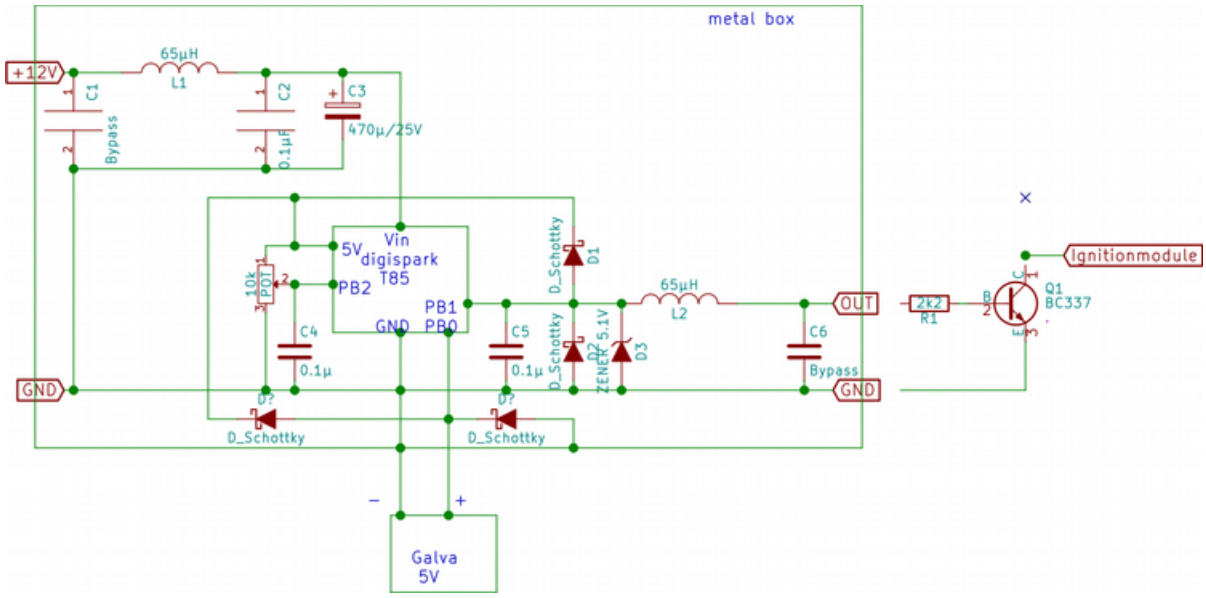


The sockets on the top were used by the previous arduino nano USB boards (I burnt 3 of them, by insufficient protection) the digispark is on the bottom left.



The test bench
the battery has a fuse in the wiring, safety first !

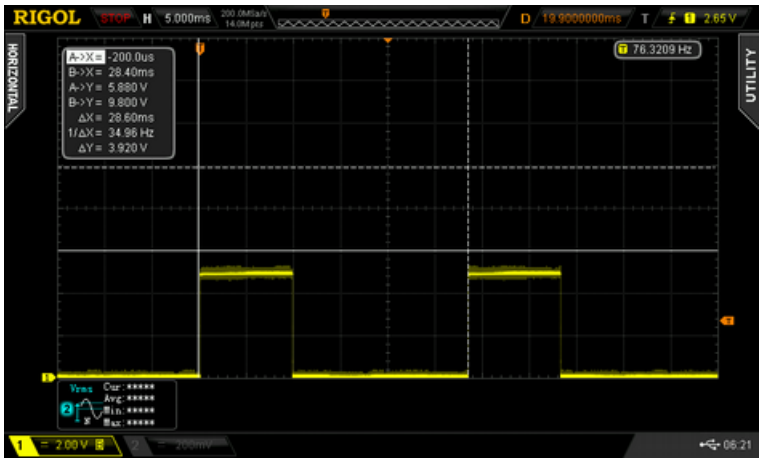
6.1 generator schematics:



connected to the « POINTS » input of ignition module
NEVER try to connect directly to a coil

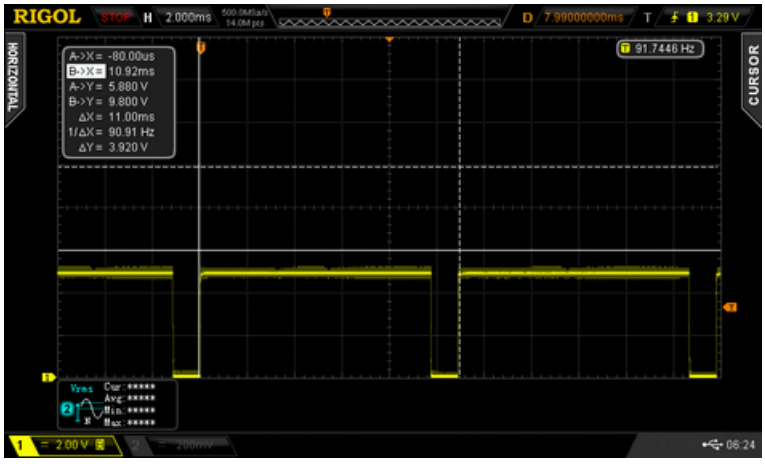
6.2 Results

dwell @ 35Hz :



magnetization time
= output at 5V
is 10ms

dwell @ 100Hz :



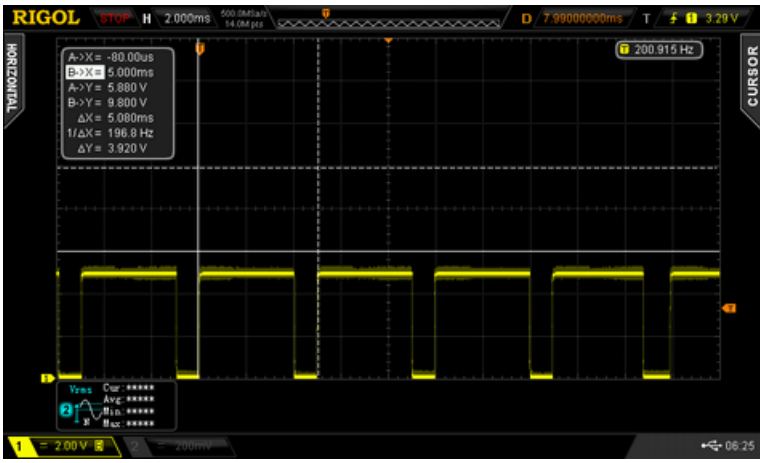
magnetization time

still around 10ms
begins to reduce

dwell @ 200Hz :

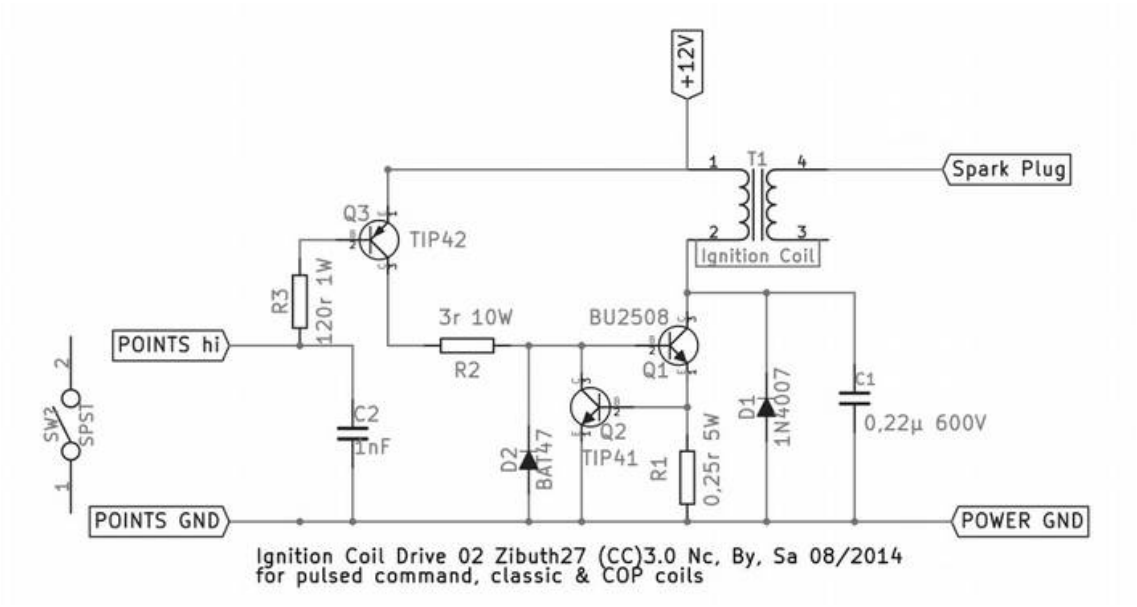
magnetization time

now 4ms



The output is positive logic and needs a transistor to invert the logic to negative and emulating the points. It's a logic signal, not a power logic : does not support the current from a coil and needs an electronic ignition module.

6.3 The ignition driver module :



http://www.hackerschicken.eu/www/electric/commande_allumage.pdf

7 Program

The program skeleton :

```
/* tiny85 digispark ignition pulse generator
 *
 * Zibuth27
 *
 * status: OK
 * keywords: potentiometer ratiometric, PWM: frequency linearized
 * automatic dwell = 10ms except if spark time (1ms) mandates to reduce it
 *
 * pin assignment
 * PB0 pin5 OC0A OC1A/ = rpm analog galvanometer
 * PB1 pin6 OC0B OC1A  = ignition pulse
 * PB2 pin7 PINB2 ADC1 = potentiometer rpm control
 * PB3 pin2 ADC3
 * PB4 pin3 ADC2
 * PB5 pin1 RST        = do not use
 *
 * RMZ#241
 * fuses lf = F1, hf = DD, ef = Fe
 *
 * 2017/08/30
 */

#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint32_t ticks;

void main (void)
{
  // ports
  DDRB = 0x03;

  // timer0

  // timer1 preset values, frequency & ratio will be changed on run


  // ADC converter


  while (1)
    {
                      // select rpm control capture ADC1
                      // start conversion
                      // wait for conversion complete

                      // analog display on galvanometer PB0
                      //  display limit because disp is 16 bits

                      // magnetization time
                      // (real goal of dwell)
                      // spark duration 1ms

                      // normal 100ms, min 4ms and guard for spark time 1ms

    } // while
} // main

ISR (TIMER1_OVF_vect)
```

8 Conclusions on the Digispark based generator

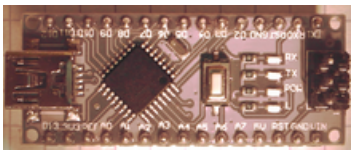
This device is easy to build and simple.

Frequency 5 to 200Hz, dwell 10ms, reduces automatically to have a spark time of 1ms, in all conditions.

9 Version II

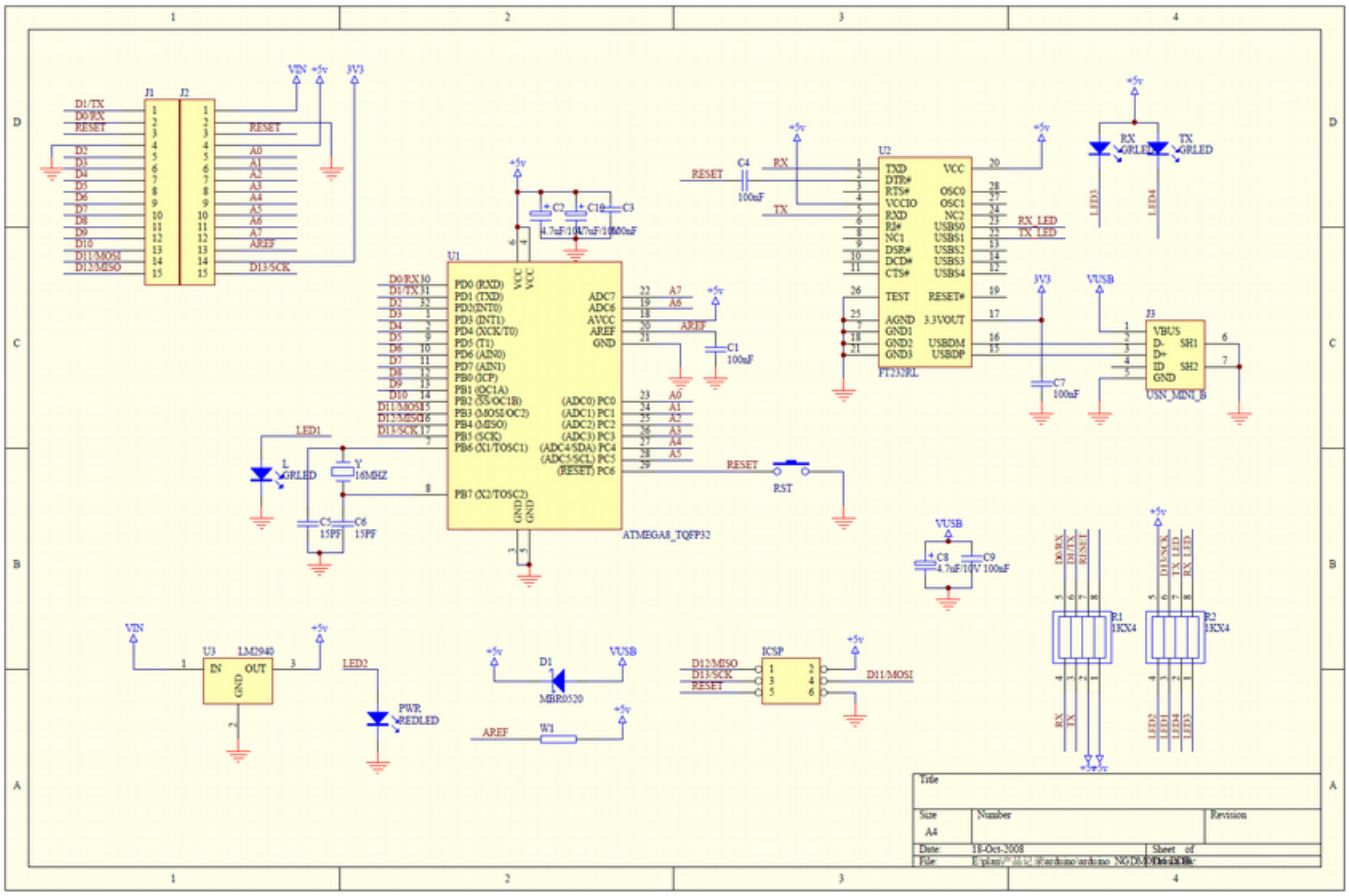
This second version is built on an **arduino nano USB**, slightly bigger.
This is no longer a KISS version, and I do not want it to be freely spreaded as the version 1. The time base is more accurate. The tiny85 has a 10% accuracy, reduced to approx 1% with the non-obvious chip calibration procedure, while the arduino nano, with a ceramic resonator has an intrinsic accuracy of 0,5% (quartz oscillator, when available, is 0,001%). The tiny85 can only run at 8MHz while the m328p (the µC inside the arduino nano) has a 16MHz clock (provided you change the lfuse, clocking normally at 2MHz), good enough for this application as the version II request more logic operations. The m328p has more available I/Os and I need additional ports : buttons and LEDs. You have to take care that the reference voltage for ADC converter has to be selected: external Avcc. I choose it also because it has a convenient built-in ICSP port.

Board is 18 x 45 mm



For using the ceramic resonator at 16 MHz, be sure the lfuse is set at 0XE7

9.1 schematics



9.2 Internal ressources

Timer1: 16-bit
The internal time base is a tick variable, incremented by timer1 OVF interruptions, adjusted by ICR1 register. The tick occurs at 10kHz (0,1ms) because I like exact variables, and their easy check by a scope.
COM1A1 = no waveform.
Mode fast PWM
frequency = 10,033kHz

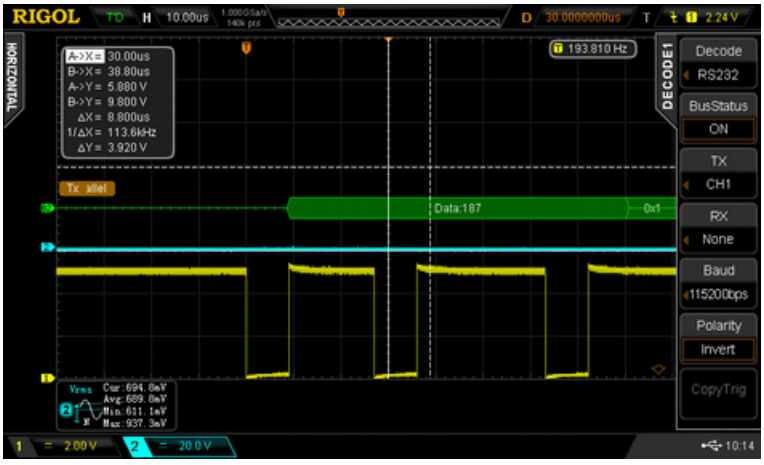
timer2: 8-bit
for analog display (galvanometer)
the timer2 has a bigger prescaler than timer0, well suited for LF output
COM2A1 = frequency
mode fast PWM
frequency = 61Hz enough for the needle to appear stable, the PWM is 8-bits

usart
used at 1Mbps for debug

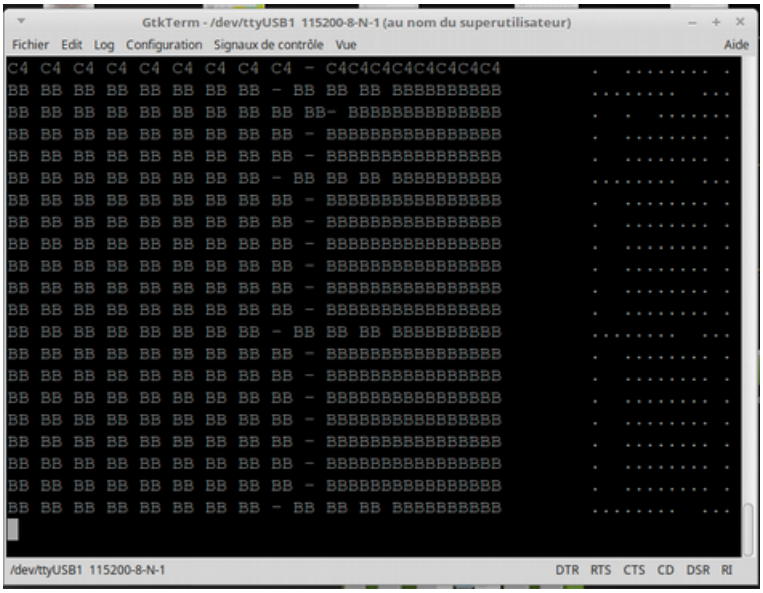
9.3 Development tools

For helping the development, when I need to know the value of internal variables, I use the RS232 feature of the chip (does nor exist in the tiny85, you need to emulate the hardware, much slower). As it is hardware into the chip, it goes faster than the software counterpart.
Example with a 155Hz command
the scope can read 187 in decimal (he can also read in hex 0xBB, a very good tool this Rigol DS2071!)

it shows also the errors parity/frame in red, if any



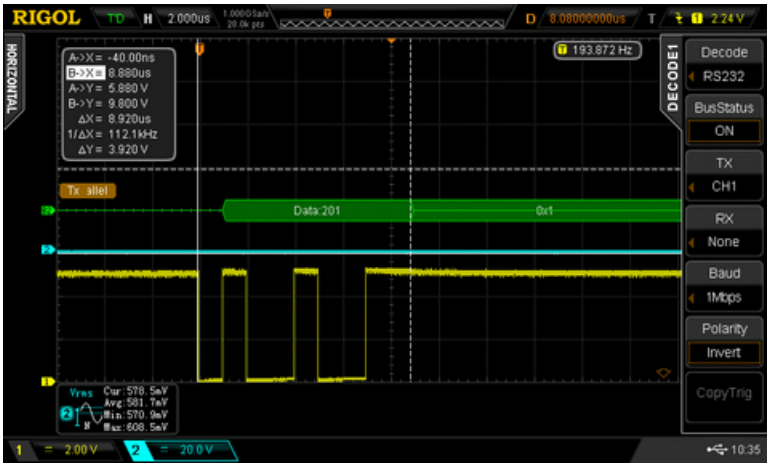
I can also use a PC Terminal program like GtkTerm with the help of a TTL RS232-USB “cable”



Gtkterm is restricted to 115200 bps (showed on the top line), maybe the hardware-software link restricts too
it reads only ASCII or hexadecimal

Because the RS232 is hardware created, and the µC time base is a ceramic 16MHz, I can use the speed of 115200 bps, total duration of the transmission is 80µs, transparent in this use.
The hardware of the µC can transmit at **1Mbps**, the PC link cannot. **The total transmission of a character is then less than 10µs**, inducing practically no overhead in the program, it allows the transmission during real working of the program.

At 1 megabits/s
Here reading of 201
= 0xC9, approx 165Hz

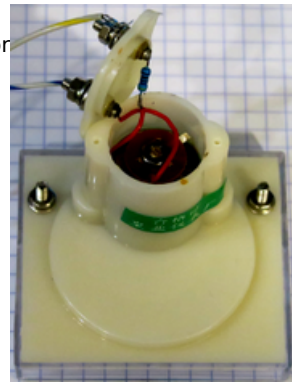


The ability to read in decimal is very convenient, when estimating analog data!
You need a pilot license (which fortunately I made, but for single engine piston only!) for managing this scope. I actually did not pass all the qualifications possible for this race engine.

9.4 Galvanometer

A cheap model is OK. I use a 5v version, as it has already a calibration resistance. But sometimes, the internal AMS1117 (equivalent to LM2940) lo-dropout regulator, has a voltage somewhat low, here Vcc = 4,9740V with a 6-digit table multimeter. The combination Vcc-voltmeter, and the quality of calibration show that the maximum of the needle does not go right enough for displaying 200Hz. A simple resistor In parallel with the internal one corrects it, or you may prefer to redraw the scale, up to you.

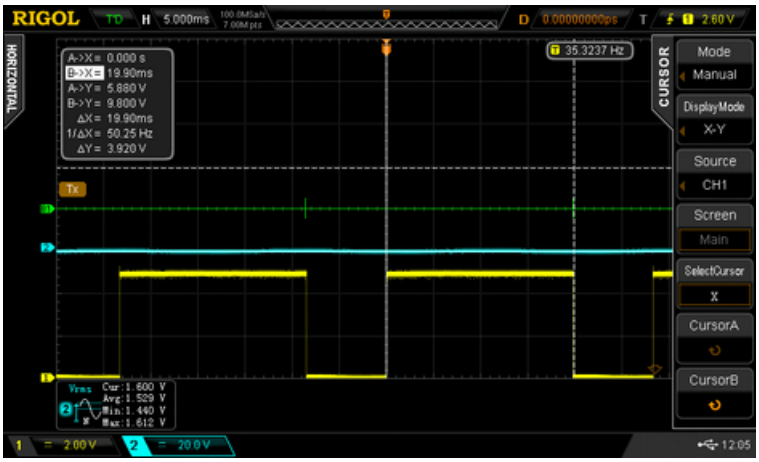
The internal calibration resistor ⇒



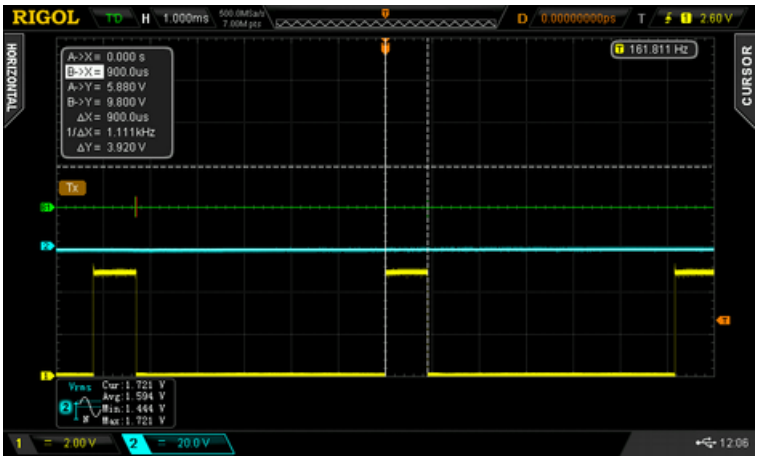
The galvanometer draws 1,0036mA at 0,171V (internal resistance of 170Ω).

9.5 Results

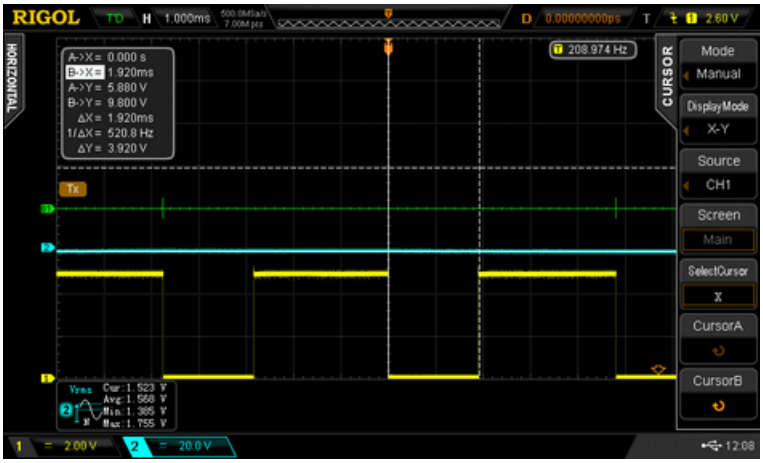
Dwell adjusted to max



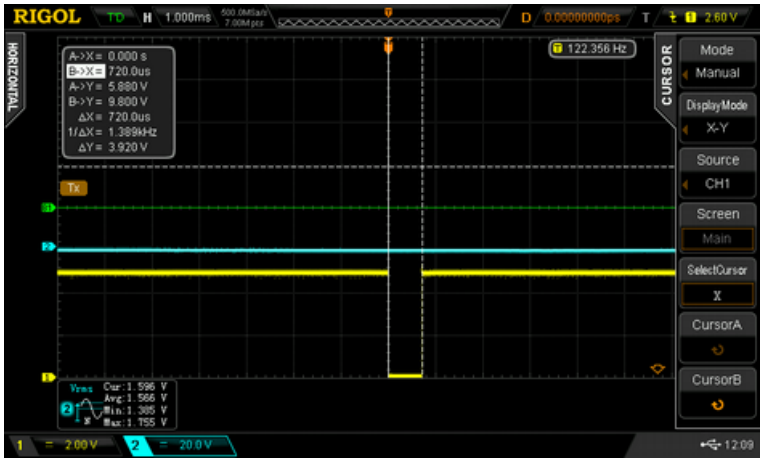
dwell adjusted to min



spark adjusted to max



spark adjusted to min



these measurements are made with output in positive logic, to be connected to an ignition module, as points emulation via an inversion/separation transistor, never connect to a coil !

10 Conclusions for version II

The version II is now able to generate pulses from 5Hz to 200Hz. The spark time can be adjusted from 0,9ms to 2ms, 1ms at reset. The dwell time can be adjusted from 1 to 20ms, 10ms at reset. Not released yet, some improvements still to be made. This is a tool for developers, version I is enough for normal users.

